Published in the proceedings of the Seventh Workshop on Enabling Technologies, (WET ICE '98), IEEE Computer Society Press, 1998.

Efficient Security for Large and Dynamic Multicast Groups

Germano Caronni†, Marcel Waldvogel‡, Dan Sun‡, Bernhard Plattner‡

†Sun Microsystems Inc., Internet Commerce and Security, Palo Alto, USA, gec@acm.org*

‡Computer Engineering and Networks Laboratory (TIK), ETH Zürich, Switzerland, {waldvogel,sun,plattner}@tik.ee.ethz.ch

Abstract

Proposals for multicast security that have been published so far are complex, often require trust in network components or are inefficient. In this paper we propose a series of novel approaches for achieving scalable security in IP multicast, providing privacy and authentication on a group-wide basis. They can be employed to efficiently secure multi-party applications where members of highly dynamic groups of arbitrary size may participate.

Supporting dynamic groups implies that newly joining members must not be able to understand past group communications, and that leaving members may not follow future communications. Key changes are required for all group members when a leave or join occurs, which poses a problem if groups are large. The algorithms presented here require no trust in third parties, support either centralized or fully distributed management of keying material, and have low complexity ($O(\log N)$ or less). This grants scalability even for large groups.

Keywords: Secure multicasting, tree-based key distribution, multicast key distribution schemes, distributed key management

1 Introduction

With IP multicasting being offered in the Internet, multi-party applications are fast becoming an important class of distributed applications, as is demonstrated with the popularity of the experimental MBone multicast service and the applications it supports. Today, the most important class of applications using a multicast transport service are collaborative multimedia applications, such as vic or vat [MB94]. Many more distibuted applications may be implemented in an efficient way by taking advantage of multicast services. As an example, take those whose primary task is to distribute information to a set of receivers; stock data distribution and audio or video distribution services clearly belong to this class, as could Usenet news postings.

Like many unicast applications, most of the multi-party applications listed above will only be successful if privacy and authenticity of participants can be provided effiently. To this end, cryptographic mechanisms are deployed. Consider, for example, a stock data distribution service, which distributes its information to a large number of customers around the globe. It is obvious that only those people who have subscribed to the service should be able to receive this information. If a new customer subscribes, he should be able to receive stock data immediately, but not to understand information which was released before the time of his subscription. Conversely, a customer canceling his subscription should not be able to process information beyond the time of cancellation.

By consequence, the purpose of this paper will be to discuss key management schemes which guarantee that at each instance in time only actual group members will be in possession of the cryptographic keys needed to participate. A naive solution would be to create a new session key whenever someone joins or leaves the group, and to securely distribute the key to all members of the group, using unicast security mechanisms. However, such a solution would not scale, as it requires that the new session key be encrypted individually for each participant.

In this paper we propose a suite of novel approaches for achieving efficient security in multicast, enabling applications requiring secure multi-party communications even in highly dynamic groups of arbitrary size. Our approaches allow all group members to establish a mutually shared secret, which can be used to provide group-wide privacy, message authenticity, or any other property relying on a shared secret. Even transitions from one key management approach to another in a running system are possible. All approaches can offer perfect forward secrecy [Dif90], require only a small amount of calculations and storage from the participants, and avoid investing trust into third party components such as routers or re-broadcasters. Depending on the choosen approach, after a setup phase, unidirectional communication is sufficient to manage group membership, and no inter-participant communication may be required. Our techniques are not limited to IP multicast — they are also applicable to satellite broadcasts or connection-oriented multicast services as found in ATM[ATM95].

The paper is organized as follows: Section 2 presents related work, Section 3 will discuss the schemes and their relation, Section 4 evaluates the results and discusses impacts of security attacks. Section 5 concludes the paper and explores further work.

2 Related Work

Existing protocols for secure multicasting are limited to distribute session keys in static and/or small groups.

For dealing with the group key distribution in a large group with frequent membership changes, some good explorations have been done:

Spanning Tree [BD96] proposes the distribution of the key along a spanning tree generated between the members. It relies on trust in all members to forward the data without modification and does not handle group membership changes securely and efficiently.

Cliques The approach proposed in [STW97] is to improve the capability of a system to distribute session keys in dynamic

[0] Work started at ETH

groups, but the solution does not scale well to large groups, since the group manager has to perfrom O(n) exponentiations for each group membership change and messages get prohibitively large.

Iolus In Iolus[Mit97], a large group is decomposed into a number of subgroups, thus reducing the number of members affected by a key change due to membership changes. It relies on "relay nodes" performing admission control and packet rekeying. This not only requires full trust into these relays, but also increases the transmission delay, and does not handle relay failures gracefully.

Multicast Trees Very recently we came across two schemes for multicast key distribution that are remarkably similar to our own tree-based approach. One is by D. Wallner, E. Harder, and R. Agee, from the National Security Agency, currently only available as an expired Internet draft. The other scheme, by C. Wong, M. Gouda, and S. Lam, from the University of Texas, is scheduled to appear in SIGCOMM'98.

For a more complete list of related works, see [CWSP98]. Issues to be improved to reach our goal are scalability, reduction of computational complexity and reduction of trust in dedicated nodes (e.g. network components), and the necessity for group members to interoperate for the generation of a group-wide secret. We will now propose a new set of protocols, demonstrating the ability to successfully handle these issues in large and highly dynamic groups.

3 Secure Multicasting

In the solutions presented here, changes to the group's membership are possible with minimal involvement of dedicated nodes and group members, limiting number and size of messages and computing resources needed. The approaches cope with several properties inherent to multicast and broadcast environments: An unreliable (and in the case of IP also unordered) transmission channel, and the transmissions may be one-way, with no or only a minimal return channel, to reflect the nature of broadcast environments – likely users of secure multicasting. While third party entities such as routers or intermediate systems are entrusted with forwarding secured data, they are not allowed to gain access to actual keying material or plain-text payload.

As seen earlier, it is important to have a system which — even with large groups and frequent joins or leaves — neither is susceptible to implosion nor enables users to understand what was transmitted at times they were not part of the group, either before they joined or after they left or were expulsed. Additionally, any third party recording ongoing transmission and later capturing the secrets held by a participant must not be able to understand its recordings. This is known as "perfect forward secrecy" [Dif90]. To completely achieve this, the unicast connections also need to be set up using ephemeral secrets.

This section is organized as follows: First, the general architecture is discussed, followed by the detailed descriptions of the three key management approaches (Centralized Tree, Centralized Flat, and Distributed Flat), explaining the properties they make available to large, dynamic groups. The presented schemes cover a wide range of applications and security needs: From very tight control in the centralized approach to extreme tolerance to system and network failures in the completely distributed scheme.

3.1 Architecture

First, the common components are identified and explained, then their interactions during all the operations are shown.

3.1.1 Components

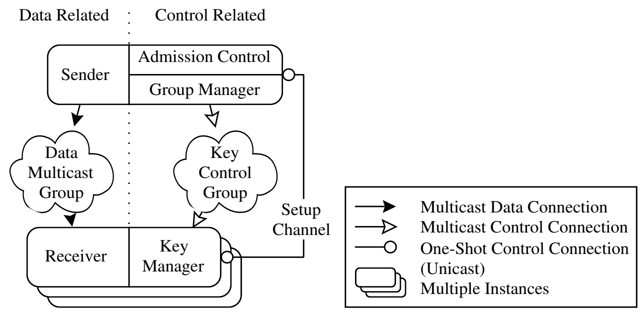


Figure 1: Secure multicasting components in a single sender, multiple recipients scenario

Figure 1 shows the basic architecture for the simplest scenario, forming the basis of the descriptions: A single sender and any number of participants (multiple sender and other scenarios will be explained below). Fundamental and common functions are explained here, while individual extensions and modifications will be pursued later. Generally, the components can be separated into two groups: (1) A group of data related components, covering components very similar to those of current insecure multicast or broadcast communication architecture. It consists of the sender, recipients, and one or more Data Multicast Groups. (2) A group of control (or key management) related components, which includes all components involved in the key agreement and key exchange process.

Sender The application prepares data as it would for non-secure transmission, then encrypts (and, using a MAC, possibly authenticates) the packets using the current Traffic Encryption Key (TEK), received from the Group Manager.

Recipient Receives the data from the Data Multicast Group and decrypts it according to the TEK given by the local Key Manager. Later steps in the application data processing will not notice any differences resulting from the encryption or authentication of data.

Data Multicast Group Any multicast, broadcast, or anycast channel delivering the secured packets from the sender(s) at least to the intended receivers. It will be used to transport the bulk of the application's data.

Group Manager Receives, admits, and processes join and leave requests from participants and sends out the messages to have Key Managers perform the necessary key changes.

Admission Control Is queried by the Group Manager to find out who is to be admitted. This function can also be delegated to a human, e.g. a chairperson.

Key Manager Receives and decodes the rekeying requests from the Group Manager, passing the resulting TEK to the Receiver.

Setup Channel Join requests from new members are usually received through this unicast connection, or via another out-of-band mechanism. This channel is only needed to bootstrap a join request and to perform authentication between the new participant and the Group Manager. A single setup component might lead to implosion problems, it is thus proposed to replicate the setup component on multiple machines, and have them establish a permanent connection to the centralized access control component. In the distributed approach (presented below) setup implosion is not an issue.

Key Control Group Any multicast or broadcast channel delivering the packets from the Group Manager to at least the intended receivers. Traffic consists of new keying material which needs to be distributed to the participants Key Managers. Transmissions over this channel have to be received by every participant, which can be achieved by (1) implementing components of any reliable multicast mechanism (such as those discussed in [FJM+95, PSB+95, PTK94]), as was done in our experimental realisation of the system, or (2) performing retransmits on a regular basis with a limited history of key changes, resulting in a soft state approach. If for any reason a receiver should be unable to receive a packet in reasonable time, the fallback solution is to contact the Group Manager again.

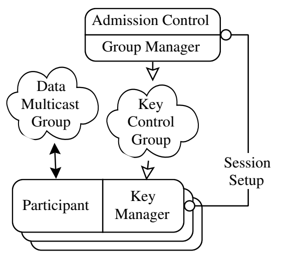


Figure 2: Group collaboration scenario

Often, there is more than one sender, and senders and receivers cannot be distinguished. Also, any receiver is free to send data encrypted or authenticated using the current TEK, and in a group collaboration environment every member of the group holds both roles at the same time, resulting in a situation as shown in Figure 2. This is a transformation of Figure 1 where sender and recipient were integrated, and the Group Manager has been isolated. All of the schemes also work in that scenario, and the distributed key management scheme even is very well suited for it. If senders and receivers are treated equally, they will be referred to using the term *participant*.

Should a unique, unmistakable, and unfakeable identification of the sender be required, as opposed to the identification as an admitted group member, it is necessary for the sender to asymmetrically authenticate each data packet. For many applications, immediate recognition of outsiders injecting traffic is crucial, but it is acceptable to detect sender impersonation by already admitted group members within a certain pre-defined time limit after the fact has occured. For these applications, it is possible to have the messages authenticated symmetrically (using a MAC) and amortize the costly asymmetric operation over several packets. To achieve this, the sender retains MAC values of all packets sent. In regular time intervals, it distributes the collected list of MAC values together with a single asymmetric signature over these MACs to the recipients. Thus, the authenticity of all the data packets sent out can be verified by the recipients with a single asymmetric operation, even if they did not get all of the original packets[1].

This procedure also can be used by the group manager to uniquely authenticate the source of keying material to the group members.

3.1.2 Basic Operations on the Group

To transmit the Traffic Encryption Key (TEK) secretly, a number of Key Encryption Keys (KEKs) are used to encrypt the control traffic containing the TEK. To distinguish the keys, each key consists of a reference tuple containing a unique ID, a version, a revision, and the keying material proper. The key to be used to decrypt a message (or part of it) is always referred to by an (ID, version, revision) tuple. The usage of independent version and revision fields allows *zero-message joining* and is explained below in the leave and join descriptions, respectively.

The abovementioned components and keys will be involved in different activities:

Group Creation The Group Manager is configured with group and access control information. Additionally, the group parameters are published using a directory service.

Single Join The new participant's Key Manager sends its request to the Group Manager, which checks whether this participant is allowed to join. If yes, the Group Manager assigns a unique ID to him, and selects a series of KEKs which will be transmitted to the newcomer. The selection of KEKs will be discussed separately for each key management scheme.

The Group Manager now increases the *revision* of all keys (TEK and KEKs) to be transmitted to the participant by passing the keying material through a one-way function (e.g. a cryptographically secure hash), then sends the keys out to the new participant. It also informs the sender(s) to update their revision and TEK. The other participants will notice the revision change from the key reference tuple in ordinary data packets, and also pass their TEK through the one-way function. Since the function is not reversible, the newcomer has no way to determine the key that was used beforehand.

Single Leave There are three ways to leave a group, namely "Silent Leave", "Voluntary Leave" and "Forced Leave". Only the third kind is of interest here as the first two do not require any action from the group manager. If the Admission Control feels a need to forcibly exclude a participant, a leave message is to be sent out. Also, participants may ask the Admission Control to exclude a member. It is up to the admission policy how to deal with such requests.

To exclude a member, all keys known to it need to be replaced with entirely new keying material. To make all remaining participants aware of this change, the key's *version* number is increased. The Group Manager sends out a message with new keying material which can be decrypted by all the remaining participants' Key Managers, but not the member which just left.

Multiple Join, Multiple Leave, Group Merge, Group Split These functions have a number of dependencies on the chosen scheme, and enhance usability of the presented archtectures. Due to space constraints, see [CWSP98] for a description.

Group Destruction The Group Manager notifies all remaining participants of the destruction, closes all network connections, destroys all keying material and frees all memory. As soon as all parties have thrown away their keying material, perfect forward secrecy covering all traffic against third party opponents is guaranteed.

3.2 Centralized, Tree-Based Key Management

Tightest control over the individual participants can be achieved by this centralized approach, which is thus suitable for applications with high security demands. It is very easy to implement and maintain, and poses very little load on the network and the receivers. All keying material is managed centrally by the Group Manager, where all joining participants have to register. To store the keying mate-

[1] This is discussed in more detail in Chapter 5 of [Car98], with application to WaveVideo[DFP97].

rial, any tree of arbitrary degree[2] can be used. The participants are represented by leaves therein. For simplicity of the explanation assume that the tree is a fully balanced, complete binary tree. The example in Figure 3 depicts such a tree with a maximum of 16 group members, and a depth of 4.

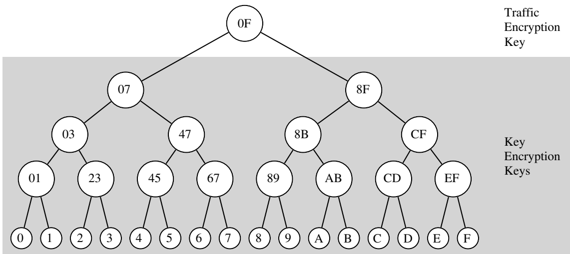


Figure 3: Binary hierarchy of keys. Labels in hexadecimal define the range of participants knowing this key.

During a setup phase, which includes admission control, each participant establishes a shared secret with the Group Manager. This shared secret is known only by the Group Manager and the individual participant, and is used as the lowest level Key Encryption Key (KEK). The Group Manager stores it in the leaf node associated with this participant, and uses it whenever a truly private communication with this participant is required — such as during the join operation. Its revision is increased after each use to insure perfect forward secrecy. The nodes in the binary tree held by the Group Manager contain further KEKs, used to achieve efficient communication of new keying material when the membership of the group changes. These nodes do not represent actual systems or intermediate entities, but only hold keys for a hierarchy of virtual sub-groups of different sizes.

Each participant holds a different subset of keys from the tree, more specifically those keys that are in the path from the participants leaf to the root node, which is used as the Traffic Encryption Key (TEK). These intermediate Key Encryption Keys are used if a message should only be understood by a part of the group, e.g. a message encrypted with KEK 47 is understood by participants $4 \ldots 7$. This enables the transmission of new keys to only a limited set of Receivers, thereby disabling others to decrypt specific messages.

Each encrypted payload and key change message includes a reference to its key's version and revision number, such that key changes and out-of-order delivery can be implicitly detected by the Receivers. Version changes are always escorted by a separate message from the Group Manager, where the new key is provided in a secure manner. Revision changes can be resolved locally.

Join On a join operation, the participant's Key Manager unicasts its request to the Group Manger, which checks with Admission Control and assigns an ID (say 4), where the participant's individual key is stored (usually the ephemeral unicast session key already employed for the join request). The participant ID is chosen such that it identifies the traversal of the tree, leading to a unique leaf, and thus determining the IDs of the keys known to the receiver. As an alternative to the explicit assignment of IDs, it is possible to use the participant's address (e.g. the IP address and port number, or a function thereof) of participants as IDs. The Group Manager increases the revision of all the keys along the path from the new leaf to the root (Key Encryption Keys 45, 47, 07 and the Traffic Encryption Key $0F$), puts them through the one-way function and sends the new revision of the keys to the joining participant, together with their associated version and revision numbers. At the same time, all senders are informed of the revision change in a preferrably reliable manner, so they start using the new TEK. The receivers will know about this change when the first data packet indicating the use of the increased revision arrives. This creates less traffic and can make the revision change more reliable.

Leave To perform a leave operation, the Group Manager sends out a message with new keying material which can only be decrypted by all the Key Managers of the remaining participants. Additionally, it frees the slot utilized by the leaving participant, making it available for reuse at the next join.

Assume C is leaving. This means that the keys it knew (Key Encryption Keys CD, CF, $8F$, and the Traffic Encryption Key $0F$) need to be viewed as compromised and have to be changed in such a way that C cannot acquire the new keys. This is done efficiently by following the tree from the leaf node corresponding to the leaving participant to the TEK stored in the root node, and encrypting the new node keys with all appropriate underlying node or leaf keys. For our example, the tree in Figure 3 shows that the new Key Encryption Key CD_{new} (replacement for CD) needs to be received by D, CF_{new} by participants D, E and F, $8F_{new}$ by $8 \ldots B, D \ldots F$, and the new Traffic Encryption Key $0F_{new}$ by every participant except C. Instead of encrypting the new keys individually for each of the intended participants, we take advantage of the existing hierarchy:

- CD_{new} is encrypted for D, the only recipient in need of it.
- CF_{new} is sent twice, each copy encrypted with one of its two children keys, the existing EF and the new CD_{new}, so it can be decrypted by the intended recipients $D \ldots F$.
- $8F_{new}$ is similarly encrypted for those knowing $8B$ or CF_{new}.
- $0F_{new}$ is finally encrypted for those holding key 07 or key $8F_{new}$.

This results in the following message being sent out:

$E_D(CD_{new})$	
$E_{EF}(CF_{new})$	$E_{CD_{new}}(CF_{new})$
$E_{8B}(8F_{new})$	$E_{CF_{new}}(8F_{new})$
$E_{07}(0F_{new})$	$E_{8F_{new}}(0F_{new})$

Along the path to the leaving node's leaf, all new keys except the bottom two rows will be encrypted for their two children. The new key in the leaver's parent node will be encrypted once. This results in $2W - 1$ keys being sent out, where W represents the depth of the hierarchy and also the length of the ID. Thus, even for a huge group with 4 billion participants ($W = 32$) and 128 bit keys, a single message of around 1200 bytes[3] multicast to everyone in the group establishes the new secrets. Processing this multicast message will require at most W decryption operations from the participants, with an average of less than 2 decryptions.

Multiple Leaves Intuitively, this can be extended to multiple leaves. The simplest and most obvious is the exclusion of a subtree, but it can be generalized to any arbitrary group of nodes. Using a single message for multiple leaves takes advantage of path overlaps, so several keys will only need to be created and sent out once per message instead of once per leave operation. This can be used to efficiently coalesce multiple leave (and join) operations into a single message.

[2]The degree of each node can possibly be different, and only the Group Manager needs to be aware of each node's degree.

[3]One Traffic Encryption Key with key id, version, and revision (each 32 bit long) encrypted for two groups, $W - 1$ Key Encryption Keys with 31 bit version and 1 bit revision encrypted for two sub-groups and one leaf Key Encryption Key, encrypted for a single node. One bit revision is enough for KEKs, since only the KEKs issued by the last leave operation must be protected from future joining participants.

Colluding participants can be reliably excluded by either sequential exclusions of them, or by grouping them together into a multiple leave operation.

Multiple Joins Similarly, if several joins happen in short succession, the revision of the TEK and the KEKs shared between the newcomers only need to be increased once, if newcomers can be allowed to decipher a small amount of data sent out before they were admitted (usually only a fraction of a second). If frequent joins are to be expected, the architecture may be changed such that the acutal senders are responsible for revision increases of the used TEK. They may increase the revision in regular, short intervals (such as half a second), thus creating a limited window for newcomers to read past traffic, but at the same time removing the need for the Group Manager to reliably keep in contact with the senders. If leaves and joins happen interleaved, they can both be grouped individually.

Group Merge, Group Split To merge two independent groups, their two trees can be joined by adding a new root node, which becomes the new TEK for the joint group. The former TEKs become the KEKs for the second level. By undoing this operation, the merged group can be split at a later point in time. To split mingled subgroups, each of the new Group Managers performs a Group Leave operation on the foreign members.

3.3 Centralized Flat Key Management ($C^\flat$)

Instead of organizing the bits of the ID in a hierarchical, tree-based fashion and distributing the keys accordingly, they can also be assigned in a flat fashion (Figure 4). This has the advantage of greatly reducing database requirements, and obviates the sender from the need of keeping information about all participants. It is now possible to exclude participants without knowing whether they were in the group in the first place.

	TEK	
ID Bit #0	KEK 0.0	KEK 0.1
ID Bit #1	KEK 1.0	KEK 1.1
ID Bit #2	KEK 2.0	KEK 2.1
ID Bit #3	KEK 3.0	KEK 3.1
	Bit's Value = 0	Bit's Value = 1

Figure 4: Flat ID assignment

The data structure held by the Group Manager is a simple table, with $2W + 1$ entries. One entry holds the current TEK, the other $2W$ slots hold Key Encryption Keys. W represents the amount of bits in the participant ID, which normally will be equal to its transport layer or network address. For each bit in the network address, two keys are available. Each participant knows W of those keys, depending on the value of the single bits in its address. All keys have associated version and revision numbers as in the tree scenario above.

The table contains $2W$ KEKs, two keys for each bit $b \in W$, corresponding to the two values $v \in \{0, 1\}$ that bit can take. The key associated with bit b having value v is referred to as K$b.v$ ("Bit Keys"). While the keys in the table could be used to generate a tree-like keying structure (e.g. by starting with the key associated with the highest-order address bit, and combining this with the key of the next level and so on, to create the shared secrets of ever diminishing subtrees), they can also be used independently of each other.

The results are very similar to the Tree-Based Control from Section 3.2, but the key space is much smaller: For an ID length of W bits, only $2W + 1$ keys (including TEK) are needed, independent of the actual number of participants. The number of participants is limited to 2^W, so a value of 32 is considered a good choice. To allow for the separation of participants residing on the same machine the ID space can be extended to 48 bits, thus including port number information. For IPv6 and calculated IDs, a value of 128 should be chosen to avoid collisions. This still keeps the number of keys and the size of change messages small. Besides reducing the storage and communication needed, this approach has the advantage that nobody needs to keep track of who is currently a member, yet the Group Manager is still able to expel an unwanted participant.

Join To join, a participant contacts the Group Manager, where it is assigned a unique ID and receives the keys corresponding to the ID's bit/value pairs, after previous revision increment. The ID may also be derived form the network address. As an example, a newcomer with (binary) ID 0010 would receive the TEK and the Key Encryption Keys K3.0, K2.0, K1.1, and K0.0 over the secure setup channel, after their revision was increased.

Leave All keys known to the leaving participant (the TEK and W KEKs) are to be considered invalid. They need to be replaced in a way intractable to the leaver, but easily computable for all remaining participants. The Group Manager sends out a multicast message consisting of two parts: Firstly, it contains a new TEK encrypted for each of the valid KEKs so that every participant with at least a single bit of difference with the leaver's ID can calculate the new TEK. Secondly, it contains a new replacement KEK encrypted with both the old KEK and the new TEK for each of the invalid KEKs, so that every participant remaining in the group can update the KEKs it previously had, but does not gain any further knowledge about the keys the other participants have. An example for the message generated when the participant with (binary) ID 0110 leaves is shown in Figure 5.

E(KEK 0.0_{new})	$E_{KEK\ 0.1}$(TEK)	ID Bit #0
$E_{KEK\ 1.0}$(TEK)	E(KEK 1.1_{new})	ID Bit #1
$E_{KEK\ 2.0}$(TEK)	E(KEK 2.1_{new})	ID Bit #2
E(KEK 3.0_{new})	$E_{KEK\ 3.1}$(TEK)	ID Bit #3
Bit's Value = 0	Bit's Value = 1	

The new KEKs are encrypted using a function of the old KEK and new TEK

Figure 5: Centralized Flat: Message to exclude participant 0110

Expelling Multiple Colluding Participants Note that — unlike in the Centralized Tree approach — expelling colluding participants can not easily be done in the flat approach. Here, they can share their key tables, and thus cover a subgroup defined by the KEKs they do not have in common. Every participant sharing each of his individual KEKs with at least one of the colluding parties is indistinguishable from them in terms of keying material that he holds. Most other approaches known to us are unable to exclude colluding participants — short of re-creating the whole group without them. With the Centralized Flat approach, excluding colluding participants is possible by overspecifying the range, i.e. considering all keys held by the colluding participants to be tainted. This will usually exclude a certain amount of valid participants as well, and they will have to re-register with the group manager.

The minimal number of colluding users needed until they can only be expelled by group re-creation ("resistant") is not limited to two, but can be increased to any arbitrary number. For simplicity, the scheme has been described in terms of bits, but can be generalized to *symbols* with any number of values V, e.g. by combining several bits into one symbol. For the same size ID, this will reduce the number of symbols W and thus the number of keys each participant will hold. At the same time, this will increase the number of keys a colluding group needs to hold to V per symbol, requiring at least V conspirators with carefully chosen IDs to become resistant.

Increasing V has the drawback that more storage is needed at the Group Manager (the Participants are not affected). So at group creation time, V should be selected according to the expected conspiracy risk and the cost of re-creating the group or re-joining participants which were accidentally excluded by overspecifying the range.

3.4 Distributed Flat Key Management ($D^\flat$)

The main concerns with centralized approaches is the danger of implosion and the existence of a single point of failure. It is thus attractive to search for a distributed solution for the key management problem. This solution was found in completely distributing the key database of the Centralized Flat approach, such that all participants are created equal and nobody has complete knowledge. As in the Centralized Flat approach above, each participant only holds keys matching its ID, and the collaboration of multiple participants is required to propagate changes to the whole group. There is no dedicated Group Manager, instead, every participant may perform admission control operations.

Since there is no Group Manager knowing about the IDs in use, the IDs need to be generated uniquely in a distributed way. Apparent solutions would be to use the participant's network address directly or to apply a collision-free hash function.

This scheme is the most resilient to network or node failures because of its inherent self-healing capability, but is also more vulnerable to inside attacks than the others. It offers the same security to break-in attacks as the schemes discussed above; thanks to its higher resilience to failures, it can be considered stronger against active attacks.

First Participant The first participant in the group will find that no heartbeat exists and start to create its own keys (the TEK and W of the $2W$ KEKs), the ones it would have received from the Group Manager in the Centralized Flat scheme. Then it starts a heartbeat announcing itself and the fact that it is Key Holder for the keys it just generated.

Join All further joins will see the heartbeat and select a previous participant (from the sender address of packets, the list of key creators from the heartbeat, or expanding multicast rings) who is willing to admit them.[4] This *introducer* will send the newcomer the keys the two of them share (the TEK and the applicable KEKs, all with increased revision). KEKs which are needed by the newcomer and do not already exist, are created as in the initial operation. Since the ID can be calculated from the network address, it is easy to select participants having the remaining keys (the introducer, having more knowledge about the group, can assist the newcomer).[5]

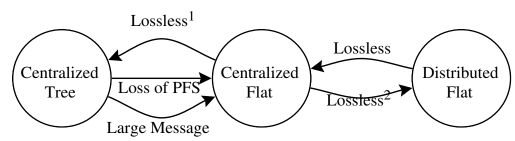
Leave The leave operation works analogous to the description in Section 3.3, with the participant taking care of someone's leave ("excluder") becoming Key Holder of this new version, announcing the new key and who has left (to update the other participants' Admission Control). Since the excluder will not know all keys whose version needs to be increased, the current Key Holder of these Keys will perform the version increase; it works as a "key relay". Participants wishing to leave also can initiate this operation through a key relay (without supplying the new keying material, which they are not supposed to know).

The other operations such as multiple joins and leaves and group merges can be performed analogous to the description in Section 3.3 when making use of the relays, since no participant is supposed to know more than its share of keys.

Above description on Distributed Flat Key Management limits itself to the utmost necessities. For a treatment of additional details and necessary considerations, the reader is kindly referred to [CWSP98][6].

3.5 Transitions

As we have seen, the three schemes are closely related. It is thus worth exploring the possiblities to change between the schemes at run-time. The possible transitions are show in Figure 6.



¹ No security gain for old participants: Colluding old participants still cannot be expelled, participants joining after the transition can.
² Previous Group Manager still knows all keys and cannot be expelled.

Figure 6: Transitions between the three schemes

The transitions between the two flat schemes are simple, because they use the same data structure. Towards the centralized flat approach, the transition happens by appointing a new Group Manager and giving him all the keys, in the other direction it can be done even after the Group Manager ceases to exist, and can thus also be viewed as a backup solution or to create a basis to elect a new Group Manager. Its only requirement is that each participant must be able to perform access control functions, or needs to trust another participant in doing this.

This transition pair is most attractive because a heterogeneous approach combining the advantages of both schemes can easily be created: Centralized Flat is used whenever possible to simplify the participants' operation, except when the Group Manager gets overloaded or becomes dysfunctional.

The transition between the two centralized schemes is more complex, as it involves changes in the key structure. A hierarchy can be generated from the flat table in the way described in Section 3.3's Multiple Leave. Keys derived from this hierarchy are then used to populate the tree data structure held by the Group Manager.

The transition from Centralized Tree to Centralized Flat is more difficult, and depends on the internal design of the keying material generator in the Group Manager. If the keying material is generated such that perfect forward secrecy of the system is assured, a transition basically involves the notification of each participant, carrying his new keying material. Buf if a limited amount of perfect forward

[4] Of course, the newcomer has to make sure that the introducer is trustworthy, i.e. both sides perform access control

[5] These additional key contributors can perform a simplified access control procedure if the newcomer includes a MAC with the TEK

[6] This also includes an efficient, almost message-free protocol to obtain consensus on contradicting key change requests.

secrecy is sufficient, another generation process can be utilized instead. Here, the Group Manager holds $2W$ generating secrets, one for each branching on each level of the tree. A short multicast message from the Group Manager to the participants is then sufficient to reveal the generating secrets to those entitled to undestand them, and leads to the table data structure. Finding a solution to this which retains perfect forward secrecy is under investigation.

4 Evaluation

The three presented schemes behave differently in terms of offered functionality, achieved performance, and how they deal with security threats. These properties will now be explored.

4.1 Offered Functionality

Table 1 compares the properties for each scheme. Most properties are self-explanatory, the others are described here:

Multiple leaves Multiple leaves are more difficult in the approaches using flat datastructures. Having multiple invalidated fields causes the table to become sparse, thus the normal mechanisms can not be used. Forcing out collaborating entities is difficult.

Easily recoverable If the group manager or other group members suddenly disappear, the flat approaches can recover from this situation by either electing a new group manager in the centralized approach, or shifting key holders in the distributed approach. This does not involve the cooperation of the whole group, but only a few participants. Thus failure recovery or self-healing can be achieved.

Assigned IDs While the centralized flat approach can work with assigned IDs, it may be unwanted to remember the assignment of IDs, and thus the use of IDs defined by the network (or a function thereof) may be preferred.

Exclusion of colluding participants This is possible in the Flat schemes, but will also exclude a number of valid participants, which will need to join again.

Property	Tree	$C^\flat$	$D^\flat$
Allows establishment of group-wise key to achieve privacy and/or authenticity	yes	yes	yes
Perfect forward secrecy	yes	yes	yes
Dynamic join and leave can be handled	yes	yes	yes
Trust in third parties required	no	no	no
Designed for one central controlling entity	yes	yes	no
Controlling entity must know all participants	yes	no	no
Multiple leaves	yes	diff	diff
Exclusion of colluding participants	yes	diff	diff
Joining and separation of groups	easy	yes	yes
Setup implosion is an issue	yes	yes	no
Return channel required during operation	no	no	yes
Assigned IDs or Network IDs	both	both	net
Single point of failure	yes	yes	no
Easily recoverable	no	yes	yes
Small database	no	yes	yes
Involvement of multiple parties for leave/join	no	no	yes

Table 1: Properties of different schemes (diff=difficult)

4.2 Useability

While the centralized approaches are better suited for broadcasting and high-security applications, the distributed approach fits more into dynamic conferencing without a dedicated session chair. While memory requirements for the group manager are significantly higher in the tree scenario (see memory consumption below), this allows for an additional level of control, and may thus be necessary anyway, and worth its cost in certain applications.

The multitude of available features, such as perfect forward secrecy, self-healing, no need for participants to cooperate or return channels to the manager, the possibility to make a transition form one scheme to the other, migrate control and no required trust in third parties allow these approaches to fulfill many different basic needs. They compare favorably to existing approaches in terms of simplicity, reliability, computational requirements and achieved security.

4.3 Achieved Performance

Ressource usage is a critical point in all applications that offer cryptographic functions. Relevant costs (both for the group manager and the participants) are:

- CPU consumption
- Memory consumption
- Communication bandwith
- Typical end-to-end operation delay

The tree-based part of the system has been implemented. In view of the simplicity of the presented architecture, a sound assessment of the involved costs can be made. The upper bounds given as concrete values are so far confirmed by our implementation, and are appropriate for a Sun "Ultra 1" workstation. Due to the ever recurring space constraints of this publication, exact numbers and costs of operations as related to the presented approaches can be found in [CWSP98].

Memory consumption is very different in the tree vs. flat scenarios. For the tree, the group manager needs to hold all N participants, and an additional $N - 1$ KEK nodes. This corresponds to a storage of about 40 bytes per tree node or leaf, in an uncompressed tree, or two times this figure for each prospective participant. The tree can be sparsely populated and compressed. It can also be grown at run-time, so the group manager need not commit to a certain size in the beginning. In the tree scenario, memory requirements for each participant amount to W times 40 bytes, or less than 10kB even for IPv6 IDs. In the flat scenarios, the memory requirement for each participant and the Group Manager is small. Some additional information may need storage, such as key ownership, but total cost is below 20kB in all cases. This makes the approach usable on platforms with comparatively reduced ressources, such as embedded systems.

On the communication side, join operations in centralized scenarios induce no additional traffic, and participants are notified of key revision changes implicitly, by the reception of messages encrypted with a higher revision number. A leave operation causes a message of typically $2W * 40$ bytes to be sent, or about 1-2 kB. This message may need to be retransmitted in one of the reliable multicast implemenations, increasing the pariticpants delay until he receives the updated keying material. In the distributed scenario, multiple exchanges are required, resulting into $2W$ multicast messages in the worst case. This may also involve a few unicast messages to cover gaps between unrelated subgroups.

4.4 Co-operation

This approach builds a complete framework, but it doesn't stand alone. It works nicely on top of (unicast) security architectures such as the one mandatory for IP version 6 [Atk95]. We are working on an integration into SKIP [CLA+96], which is available in source for a number of platforms.

Our schemes also work atop any reliable multicast protocol (e.g. [FJM+95, PSB+95, PTK94]). It can also work without any, but this will increase the load on the Group Manager. It also can take advantage of the proposed Integrated Services architecture for the Internet and the associated resource reservation protocol, RSVP [BCS93] to limit the work that has to be done by the reliable multicast protocol and thus reduce latencies.

Our schemes do not rely on specific cryptographic algorithms and protocol, but can use any of a number of them providing a basic functionality. So even if one of them should have to be considered weak or even broken, these components are easy to change and this framework will continue to work.

5 Conclusions and Further Work

In this paper we presented a complete framework for secure multicasting. The core of the framework consists of three approaches which have different properties, but rely on the same basic philosophy. All our approaches organize the space of keys that will eventually to be assigned to group members in a unique way, without actually generating the keys as before they are needed. Only when new group keys need to be established, they are generated and distributed to only the members of the group affected by a change. Our organization of the key space assures that all operations on groups may be executed with a complexity of $O(logN)$ or less, where N is the size of the group, and the complexity is measured in the size and number of messages exchanged, and the number of cryptographic operations to be performed by any of the participants.

Our three approaches differ in some important aspects. Among others, they offer the system designer a choice between

- centralized or distributed key management,
- no or some trust in other participants,
- varying degrees of load on the participants, and
- tight control of the group or failsafe distributed operation.

As discussed in the introductory section, various authors have published work on secure multicasting schemes. Some of the properties as presented in Table 1 are also offered by their approaches, but we are not aware of any scheme that has all these properties while maintaining the efficiency of ours.

Some considerations deserve further studies. Although a preliminary implementation is available and working, we still lack experiments with large and distributed groups; to this end, the integration of our experimental software into currently available platforms is planned, such as SKIP [CLA+96] and ISAKMP/Oakley [Orm97]. The possibility of a hot switching between the approaches presented as discussed in Section 3.5 is a recent discovery, and needs to be considered in detail. Specifically, an efficient translation algorithm between the tree (Section 3.2) and the flat data structure (Section 3.3) needs to be found and analyzed. Furthermore, we anticipate that batching of leave operations may be made more efficient with optimal grouping of the participants leaving within some time interval.

Acknowledgments

We would like to thank Nathalie Weiler, Radia Perlman, and the anonymous reviewers for the valuable feedback in improving our paper.

References

[Atk95] R. Atkinson. Security architecture for the internet protocol. RFC 1825, August 1995.

[ATM95] ATM Forum. *UNI Signalling 4.0*, 1995.

[BCS93] R. Braden, D. Clark, and S. Shenker. RSVP: A new resource reservation protocol. *IEEE Network*, September 1993.

[BD96] M. Burmester and Y.G. Desmedt. Efficient and secure conference-key distribution. In *Security Protocols Workshop*, pages 119–129, 1996.

[Car98] Germano Caronni. *Dynamic Security in Communication Systems*. PhD thesis, ETH Zürich, 1998. Work in progress.

[CLA+96] G. Caronni, H. Lubich, A. Aziz, T. Markson, and R. Skrenta. Skip: Securing the internet. In *Proceedings of the IEEE Fifth Workshop on Enabling Technologies (WET ICE)*, 1996.

[CWSP98] Germano Caronni, Marcel Waldvogel, Dan Sun, and Bernhard Plattner. Efficient security for large and dynamic multicast groups. TIK Technical Report TIK-41, TIK, ETH Zürich, February 1998. Also available as ftp://ftp.tik.ee.ethz.ch/pub/publications/TIK-Report41.ps.gz.

[DFP97] M. Dasen, G. Fankhauser, and B. Plattner. An error-tolerant, scalable video stream encoding and compression for mobile computing. In *ACTS Mobile Summit 1997*, volume 2, pages 762–771, November 1997.

[Dif90] W. Diffie. Authenticated key exchange and secure interactive communication. In *Proceedings of "8th Worldwide Congress on Computer and Communications Security and Protection" SECURICOM 90*, pages 300–306, 1990.

[FJM+95] S. Floyd, V. Jacobson, S. McCanne, L. Zhang, and C. Liu. A reliable multicast framework for light-weight sessions and application level framing. In *Proceedings of ACM SIGCOMM '95*, pages 342–356, September 1995.

[MB94] M.R. Macedonia and D.P. Brutzman. MBone provides audio and video across the internet. *IEEE Computer*, 27(4):30–36, April 1994.

[Mit97] S. Mittra. Iolus: A framework for scalable secure multicasting. In *Proceedings of ACM SIGCOMM '97*, pages 277–288, September 1997.

[Orm97] H.K. Orman. The OAKLEY key determination protocol. Internet Draft (work in progress), draft-ipsec-ietf-oakley-02.txt, 1997.

[PSB+95] S. Paul, K. Sabnani, R. Buskens, S. Muhammad, J. Lin, and S. Bhattacharyya. RMTP: A reliable multicast transport protocol for high-speed networks. In *Proceedings of the Tenth Annual IEEE Workshop on Computer Communications*, September 1995.

[PTK94] S. Pingali, D. Towsley, and J. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *Proceedings of SIGMETRICS '94*, 1994.

[STW97] M. Steiner, G. Tsudik, and M. Waidner. Cliques: A protocol suite for key agreement in dynamic groups. Research Report RZ 2984 (#93030), IBM Zürich Research Lab, December 1997.